1 Eksamen Høst 2020

1.1 Oppgave 1

Hva er kjøretiden til denne koden?

```
public double interestOnLoan(double amount, int n) {
    amount = amount * 1.01;
}
return amount
```

Svar.

$$O(n)$$

1.2 Oppgave 2

Hva er kjøretiden til denne koden?

```
public static int countOneBits(int n) {
    int bits = 0;
    while (n > 0) {
        if (n % 2 == 1) {
            bits++;
        }
        n = n / 2;
    }
    return bits;
}
```

Svar.

$$O(\log n)$$

1.3 Oppgave 3

Hva er kjøretiden til denne koden?

```
public static int countSteps (int n) {
    int pow = 2;
    int steps = 0;

    for(int i = 0; i < n; i++) { // n
        if (i == pow) { // n
            pow *= 2;
            for(int j = 0; j < n; j++) {
                steps++;
            }
        } else {
            steps++;
        }
    }
    return steps;
```

```
}
```

Svar.

$$O\left(n \log n\right)$$

1.4 Oppgave 4

Hva er kjøretiden til denne koden?

```
public staic String makeRandomString(int n) {
    String ans = "";
    for (int i = 0; i < n; i++) {
        char c = (char) ('a'+26*Math.random());
        ans += c;
    }
    return ans;

}
```

Svar.

$$O(n^2)$$

1.5 Oppgave 5

Hva er kjøretiden til denne koden?

```
public static double computeAreaUnderCurve(LinkedList<Double> y) {
    Double area = 0.0;

    for(int i = 1; i < y.size(); i++) {
        area = area + (y.get(i - 1) + y.get(i)) / 2;
    }
    return area;
}
```

Svar.

$$O(n^2)$$

1.6 Oppgave 6

Hva er kjøretiden til denne koden?

```
// n = list.size()
public static Collection<Integer> findLargestK(ArrayList<Integer> list, int k) {
    PriorityQueue<Integer> pq = new PriorityQueue<Integer>();
    for(int num: list) { // n
        if(pq.size() < k || pq.peek() < num) { // 1
            pq.add(num) // log k
```

```
        }

        if(pq.size() > k) { // 1
            pq.poll(); // log k
        }
    }
    return pq;
}
```

Har kan du anta at $k < n$.

> **Svar.**
>
> $$O\left(n \log\left(k\right)\right)$$

1.7 Oppgave 7

Denne koden kalles når `words` er tom og vil da fylle inn settet `words`. Hva er kjøretiden til denne koden?

```
HashSet<String> dictionary;
HashSet<String> words;

// n = dictionary.size()
// k = letters.length()

private void possibleWords(String word, String letters) {
    if(!word.isBlank() && dictionary.contains(word)) { // n
        words.add(word); // 1
    }

    int k = letters.length(); // 1

    for(int i = 0; i < k; i ++) { // k
        // add letter to word
        String nextWord = word + letters.charAt(i); // k

        // remove letter from available letters
        String nextLetters = letters.substring(0, i);

        if (i < k - 1) {
            nextLetters = nextLetters + letters.substring(i + 1, k);
        }

        // continue searching
        possibleWords(nextWord, nextLetters);
    }
}
```

Her kan du anta at $k < n$.

> **Svar.**
>
> $$O\left(k! \cdot k\right)$$

1.8 Oppgave 8

Forklar hvordan algoritmen `QuickSort` ville sortert denne listen med tall.

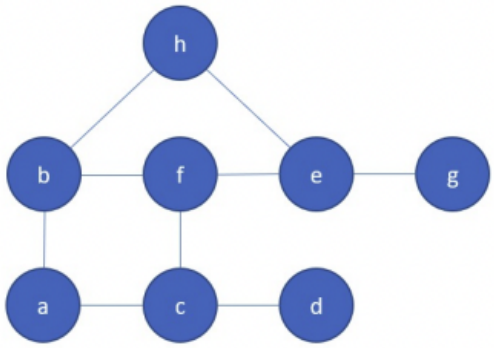
$$4, 7, 15, 1, 9, 3, 6, 12, 2$$

Svar. `QuickSort` bruker en `pivot` for å dele opp listen i verdier som er større enn og mindre enn denne verdien. Sorteringen vil skje på følgende måte: Som første pivot vil vi bruke veriden lengst til høyre i listen. Høyre og venstre er de indeksene i listen som vi foreløpig ser på, og er initalisert til 0 og lengden av listen - 1, som i dette tilfellet er 8.

Liste	Pivot	Høyre	Venstre
4,7,15,1,9,3,6,12,2	2	0	9
1,2,4,3,6,7,9,12,15	7	2	8
1,2,4,3,6,7,9,12,15	6	2	4
1,2,3,4,6,7,9,12,15	3	2	3
1,2,3,4,6,7,9,12,15	15	6	8
1,2,3,4,6,7,9,12,15	12	6	7

1.9 Oppgave 9

DFS (dybde først søk) er en metode for å besøke alle nodene i en graf. Hver node blir kun besøkt en gang. Forklar hvordan DFS kjører på denne grafen ved å stegvis beskrive hva algoritmen gjør og i hvilken rekkefølge nodene vil bli besøkt. Du skal begynne søket i node "a". Det er mange rette rekkefølger, du skal bare gi en mulig rekkefølge for DFS.

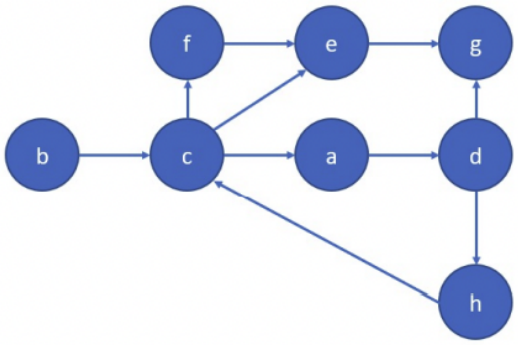


Svar. DFS opererer på *Last in, First out* prinsippet. Den vil derfor besøke alle nodene i siten til et løv, før den går videre opp i grafen og finner andre noder. Vi kan beskrive rekkefølgen og hva algoritmen gjør i en tabell.

Nåværende node	Neste node	Kø	Besøkte
a	b	c,b	
b	h	c,f,h	a
h	e	c,f,e	a,b
e	g	c,f,f,g	a,b,h
g	f	c,f,f	a,b,h,e
f	c	c,f,c	a,b,h,e,g
c	d	c,f,d	a,b,h,e,g,f
d		c,f	a,b,h,e,g,f,c
			a,b,h,e,g,f,c,d

1.10 Oppgave 10

BFS (bredde først søk) er en metode for å besøke alle nodene i en graf. Hver node blir kun besøkt en gang. Forklar hvordan BFS kjører på denne rettede grafen ved å beskrive hvilken rekkefølge nodene vil bli besøkt. Du skal begynne søket i node "a". Det er mange rette svar du skal bare gi en mulig rekkefølge for BFS.



Svar. BFS opererer på *First in, First out* prinsippet, og vi kan beskrive hva algoritmen gjør med en tabell

Nåværende node	Neste node	Kø	Besøkte
a	d	d	
d	g	g,h	a
g	h	h,c	a,d
h	c	c	a,d,g
c	f	f,a	a,d,g,h
f	e	a,e	a,d,g,h,c
e		a	a,d,g,h,c,f
			a,d,g,h,c,f,e